Tutorial - 3

Name → DIVYANSH DUBEY

Section → F

Class roll no → 55

University roll no → 2016738

Q1 Write linear search pseudocode to search an element in a sorted array with minimum comparisons.

Ans
```
for (i=0 to n)
{
    if (arr [i] == value)
        // element found
}
```

Q2 Write pseudo code for iterative & recursive insertion sort. Insertion sort is called Online Sorting. Why? What about other sorting algorithms that have been discussed?

Ans Iterative

```
Void insertion sort ( int arr[], int n)
{
    for ( int i=1 ; i<n; i++)
    {
        j=i-1;
        x = arr [i];
    while ( j>-1 && arr [j] > x)
    {
        arr [j+1] = arr [j];
        j--;
        }
        arr[j+1] = x;
      }
    }
```

Recursive

```
void insertion sort (int arr [], int n)
  {
    if (n <=1)
        return;
    insertion_sort (arr, n-1);
    int last = arr [n-1];
     int j= n-2;
    while ( j >=0 && arr[j] > last)
     {
       arr [ j+1] = arr [j];
           j--;
     }
    arr [j+1] = last;
  }
```

Insertion sort is called 'online sort' because it does not need to know anything about what values it will sort and information is requested while algorithm is running.

Other sorting algorithm:-

1) Bubble sort
2) Quick sort
3) Merge Sort
4) Selection sort
5) Heap Sort

Q3 Complexity of all sorting algorithms that has been discussed in lectures.

Ans

Sorting Algorithm	Best	Worst	Average
selection sort	$O(m^2)$	$O(n^2)$	$O(n^2)$
Bubble sort	$O(n)$	$O(n^2)$	$O(n^2)$
Insertion sort	$O(n)$	$O(n^2)$	$O(n^2)$
Heap sort	$O(n \log n)$	$O(n\log n)$	$O(n\log n)$
Quick sort	$O(n\log n)$	$O(n^2)$	$O(n\log n)$
merge sort	$O(n\log n)$	$O(n\log n)$	$O(n\log n)$

Q4 Divide all sorting algorithm into inplace | stable | online sorting.

Ans Inplace sorting

1) Bubble sort
2) Selection sort
3) Insertion sort
4) Quick sort
5) Heap sort

Stable Sorting

1) merge sort
2) Bubble sort
3) Insertion Sort
4) Count sort

Online Sorting

Insertion sort

Q5 write recursive pseudocode for binary search. What is the time complexity of linear and binary search.

Iterative

```
int b_search (int arr[], int l, int r, int key)
    {
        while (l <= r) {
        int m = ((l+r)/2);
        if (arr[m] == key)
              return m;
        else if (key < arr(m])

        r = m-1;
        else
            l=m+1;
        }
        return -1;
    }
```

Recursive

```
int b_search (int arr[], int l, int n, int key)
    {
      while (l<=r){
      int m= ((l+r)/2);
     if (key== arr[m])
           return m;
      else if (key < arr[m])
      return b_search (arr, l, mid-1, key);
        else
     return b_search (arr, mid+1, r, key);
    }
    return-1;
  }
```

Time complexity

1) Lineary search - $O(n)$
2) Binary search - $O(\log n)$

Q6 Write recurrence relation for binary recurrence relation.

Ans $T(n) = T(n/2) + 1$

$T(n/2) = T(n/4) + 1$

$T(n/4) = T(n/8) + 1$

$T(n) = T(n/2) + 1$

$T(n) = T(n/4) + 1 + 1$

$T(n) = T(n/8) + 1 + 1 + 1$

$\vdots$

$T(n/2^n) + 1$ (K times)

let $g^K = n$

$K = \log n$

$T(n) = T(n/n) + \log n$

$T(n) = T(1) + \log n$

$T(n) = O(\log n)$

Q7 Find two indexes such that A[i] + A[j] = k in minimum time complexity.

Ans
```
for ( i=0; i<n ; i++)
    {
    for ( int j=0; j<n ; j++)
    {
      if (a[i] + a[j] ==k)
        printf ("%d %d", i, j);
      }
    }
```

Q8 Which sorting is best for practical uses? Explain.

Ans Quick sort is fastest general purpose sort. In most practical situations, quicksort is the method of choice as stability is important and space available, mergesort might be best.

Q9 What do you mean by inversions in an array? Count the number of inversions in Array arr[] = { 7,21,31, 8, 10,1, 20,6,4,5} using merge sort?

Ans A pair (A[i], A[j]) is said to be inversion.

if

- A[i] > A[j]
- i<j
- Total no. of inversions in given array are 31

Q10 Selection Sort is not stable by default, but can you write a stable version of selection sort.

Ans
```
for (int i=0; i<n; i++)
    {
     int min =i;
    for ( int j =i+1; j<n ; j++)
      {
       if (a[min] > a[j])
      {
        min = j;
      }
    }
   int key = a[min];
    while [min >i)
   {
   a[min] = a[min -j];
      min --;
   }
   a[i] = key;
  }
```

Q11 Bubble sort scans array even when array is sorted. Can you modify the bubble sort so that it does not scan the whole array once it is sorted.

Ans A better version of bubble sort known as m bubble sort, includes a flag that is set of exchange made then it should be called the array is already in order because no 2 elements were switched.

```
Void bubble (int arr[], int n)
{
  for (int i=0; i<n; i++)
  {
     int swaps = 0;
  for (int j=0; j<n-i-j; j++)
    {
     if (arr[j] > arr[j+1])
    {
      int t = arr[j];
      arr[j] = arr[j+1];
      arr[j] = t;
      swap++;
    }
  }
  if (swap == 0)
     break;
   }
}
```